1   Background of the Invention

2

3   *1.   Field of Invention*

4

5   This invention relates to data storage systems.

6

7   *2.   Related Art*

8

9   Many computer applications need to store and retrieve information.

10   Information can be stored on hard disks, floppy disks, CD-ROMs, semiconductor RAM

11   memory and similar storage devices. Many of these storage systems are susceptible to data

12   loss of various forms including disk failures. A solution to the problem of disk failure

13   involves use of a RAID (redundant array of independent disks) system. One style of RAID

14   systems uses multiple hard drives to store parity data generated from the data drives, either

15   on a separate drive (known as the parity disk) or spread out among the multiple drives. The

16   use of multiple hard drives makes it possible to replace faulty hard drives without going off-

17   line; data contained on a drive can be rebuilt using the other data disks and the redundant

18   data contained in the parity disk. If a hard drive fails, a new hard drive can be inserted by

19   "hot-swapping" drives while on-line. The RAID system can rebuild the data on the new disk

20   using the remaining data disks and the redundant data of the parity disk. The performance of

1   a RAID system is improved by disk striping, which interleaves bytes or groups of bytes

2   across multiple drives, so more than one disk is reading and writing simultaneously. Files

3   are broken into chunks of data known as file blocks and these file blocks are stored in one or

4   more physical sectors of one or more hard disks. Each file block is a given size such as

5   4,096-bytes that takes up 8 sectors.

6

7   A first known problem with storage devices is that they are susceptible to data

8   corruption. This data corruption includes bit flips, misdirected I/O, lost I/O, sector shifts,

9   and block shifts. One style of RAID uses parity data to determine whether there has been

10 corruption of some data included in a disk stripe. Parity is checked by comparing the parity

11 value stored on disk against the parity values computed in memory. Parity is computed by

12 taking the exclusive-OR (henceforth "XOR") of the blocks in the data stripe. If the stored

13 and computed values of parity are not the same, the data may be corrupt. If a single disk

14 block is incorrect, the RAID system includes enough data to restore the corrupted block by

15 recalculating the corrupted data using the parity data and the remaining data in the data

16 stripe. However, such RAID systems can not determine which disk includes the corrupt data

17 from parity values alone. Although parity data is useful in determining whether corruption

18 has occurred, it does not include enough information to restore the corrupted data.

19 Moreover, it is unclear which data has been corrupted.

20

1   Checksums are another form of redundant data that can be written to individual

2   disks. The combination of parity bits across the disks along with checksums and their

3   associated information may include enough information so that the corrupted data can be

4   restored in RAID and other redundant systems.

5

6   A second known problem involves using a sector checksum for each sector of

7   data. A sector checksum is generated for each collection of data that can fill a sector. The

8   data is stored in a disk sector, along with the associated sector checksum. Some known

9   systems include reformatting a collection of hard disks from standard sector sizes such as

10   512-byte sectors to include sector checksums in each sector such as reformatting to 520-byte

11   sectors. Data corruption in the disk sector can then be detected by using the sector checksum

12   because the stored checksum would not match a computed checksum. However, data

13   corruption such as sector slides, misdirected reads and writes, and lost sectors would not be

14   detected at the disk sector level. For this type of corruption, a checksum computed from the

15   sector data would match the stored checksum.

16

17   A third known problem is storing checksums in reserved locations separate

18   from the associated data. A separate read or write operation of the checksum is required for

19   every read or write operation of the associated data. This can result in performance loss in

20   some workloads.

1   Accordingly, it would be advantageous to provide an improved technique for

2   the error checking and correction of data storage systems. This is achieved in an

3   embodiment of the invention that is not subject to the drawbacks of the related art.

4

5   Summary of the Invention

6

7   The invention provides an improved method and apparatus for a reliable data

8   storage system using block level checksums appended to data blocks.

9

10   In a first aspect of the invention, a block-appended checksum is created at the

11 filesystem block level, where a block is the filesystem's unit of transfer. In a preferred

12 embodiment, the data storage system is a RAID system composed of multiple hard disk

13 drives, including a parity disk drive and a controller for the drives. Files are stored on hard

14 disks in storage blocks, including data blocks and block-appended checksums. The block-

15 appended checksum includes a checksum of the data block, a Virtual Block Number (VBN),

16 a Disk Block Number (DBN), and an embedded checksum for checking the integrity of the

17 block-appended checksum itself. The block-appended checksum reliably detects corruption

18 of data within a sector such as bit flips, as does a sector checksum. However, a block-

19 appended checksum also reliably detects data corruption across sectors including sector

20 slides, misdirected reads and writes, and lost sector I/O.

1   The combination of (1) parity bits across the RAID system stored on the parity

2   disk, (2) the remaining uncorrupted data in the data disks, and (3) block-appended checksum

3   within each disk includes sufficient information so as to enable detection and restoration of

4   corrupt data in RAID systems and other similar devices. Such a combination is preferable to

5   using block-appended checksums alone because block-appended checksums are limited to

6   detecting errors.

7

8   In a second aspect of the invention, a file system includes file blocks with

9   associated block-appended checksum to the data blocks. The file blocks with block-

10   appended checksums are written to storage blocks. In a preferred embodiment a collection

11   of disk drives are formatted with 520 bytes of data per sector instead of the more commonly

12   found 512 bytes per sector. For each 4,096-byte file block, a corresponding 64-byte block-

13   appended checksum is appended to the file block. When this is written to disk, the first 7

14   sectors includes most of the file block data while the 8th sector includes the remaining file

15   block data and the 64-byte block-appended checksum for a total of 4,160-bytes of data.

16   Because the block-appended checksums are appended to the file blocks, every read or write

17   to a storage block includes the reading or writing of the file block and the block-appended

18   checksum in a single operation. In known cases, this results in greatly improved

19   performance.

20

1   In a third aspect of the invention, I/O operations are first stored in NVRAM

2   (non-volatile random access memory). In the event of a system crash, I/O operations are

3   replayed from NVRAM, which preserves file block data. When the I/O operation is

4   performed again, the corresponding block-appended checksum information is simply

5   recalculated.

6

7   In a preferred embodiment, the invention is operative on a RAID level 4

8   system for a file server. However, in other embodiments, the invention is applicable to any

9   computer data storage system such as a database system or a store and forward system such

10   as cache or RAM.

11

12   Brief Description of the Drawings

13

14   Figure 1 shows a block diagram of a reliable, redundant data storage system

15   including block-appended checksum.

16

17   Figure 2 shows a flow diagram of a method for writing file blocks with block-

18   appended checksums to a reliable, redundant data storage system.

19

20   Figure 3 shows a flow diagram of a method for reading data from data storage

1 blocks including file blocks with block-appended checksums in a reliable, redundant data

2 storage system.

3

4 <u>Detailed Description of the Preferred Embodiment</u>

5

6 In the following description, a preferred embodiment of the invention is

7 described with regard to preferred process steps and structures. Those skilled in the art

8 would recognize after perusal of this application that embodiments of the invention can be

9 implemented using elements adapted to particular process steps and structures described

10 herein, and that implementation of the process steps and structures described herein would

11 not require undue experimentation or further invention.

12

13 <u>Incorporated Disclosures</u>

14

15 The inventions described herein can be used in conjunction with inventions

16 described in the following applications:

17

18 • Application Serial Number 09/642,063, in the name of Blake LEWIS, Ray CHEN and

19 Kayuri PATEL. filed on August 18, 2000, Express Mailing Number

20 EL524781089US, titled "Reserving File System Blocks", assigned to the same

1   assignee, attorney docket number 103.1033.01, and all pending cases claiming the

2   priority thereof.

3

4   • U.S. Patent Application Serial No. 09/642,062, in the name of Rajesh SUNDARAM,

5   Srinivasan VISWANATHAN, Alan ROWE, Steven R. KLEIMAN and John

6   EDWARDS filed August 18, 2000, Express Mail Mailing No. EL524780242US,

7   titled "Dynamic Data Space", assigned to the same assignee, filed August 18, 2000,

8   attorney docket number 103.1034.01, and all pending cases claiming the priority

9   thereof.

10

11 • Application Serial Number 09/642,066, in the name of Ray CHEN, John EDWARDS

12 and Kayuri PATEL filed on August 18. 2000, Express Mailing Number

13 EL524780256US, titled "Manipulation of Zombie Files and Evil-Twin Files",

14 assigned to the same assignee, attorney docket number 103.1047.01, and all pending

15 cases claiming the priority thereof.

16

17 • Application Serial Number 090/642,065, in the name of Doug DOUCETTE, Blake

18 LEWIS and John EDWARDS, filed August 18, 2000, Express Mailing Number

19 EL524781092US, titled "Improved Space Allocation in a Write Anywhere File

20 System", assigned to the same assignee, attorney docket number 103.1045.01, and all

1   pending cases claiming the priority thereof.

2   • Application Serial Number 09/642,061, in the name of Blake LEWIS, John

3   EDWARDS, and Srinivasan VISWANATHAN, filed on August 18, 2000, Express

4   Mailing Number EL524780239US, titled "Instant Snapshot", assigned to the same

5   assignee, attorney docket number 103.1035.01, and all pending cases claiming the

6   priority thereof.

7

8   • Application Serial Number 09/642,064, in the names of Scott SCHOENTHAL, filed

9   August 18, 2000, Express Mailing Number EL524781075US, titled "Persistent and

10 Reliable Delivery of Event Messages", assigned to the same assignee, attorney docket

11 number 103.1048.01, and all pending cases claiming the priority thereof.

12

13 *Lexicography*

14

15 As used herein, use of the following terms refer or relate to aspects of the

16 invention as described below. The general meaning of these terms is intended to be illusory

17 and in no way limiting.

18

19 • **Sector** – In general, the term "sector" refers to a physical section of a disk drive

20 including a collection of bytes, such as 512 or 520 bytes. This is the disk drive's

1   minimal unit of transfer.

2 •   **Storage block** – In general, the term "storage block" refers to a group of sectors, such

3   as 8 sectors or 4,096 bytes for 512 byte sectors or 4,160 bytes for 520 byte sectors.

4

5 •   **Data block** – In general, the term "data block" refers to a collection of bytes stored in

6   a storage block, such as 4,096 bytes with 512-byte sectors or 4,160 bytes with 520-

7   byte sectors each stored in 8 sectors.

8

9 •   **Block-appended checksum** – In general, the term "block-appended checksum" refers

10   to a collection of bytes, such as 64 bytes, which may include a checksum of a data

11   block, a Virtual Block Number (VBN), a Disk Block Number (DBN), and an

12   embedded checksum for checking the integrity of checksum information itself.

13

14 •   **Stripe** – In general, the term "stripe" refers to the collection of blocks in a volume

15   with the same DBN on each disk.

16

17 •   **Volume** – In general, the term "volume" refers to a single file system spread across

18   multiple disks and associated disk drives. Known data storage systems have current

19   size limits, such as greater than one terabyte and are included in multiple volumes.

20

1   •   **DBN (Disk Block Number)** – In general, the term "DBN" refers to a location of a

2   particular block on a disk in a volume of a file system.

3

4   •   **VBN (Volume Block Number)** – In general, the term "VBN" refers to an integer that

5   maps to a disk number and disk block number.

6

7   •   **WAFL (Write Anywhere File Layout)** – In general, a high level structure for a file

8   system that is above RAID in hierarchy and includes metadata, such as one or more

9   copies of the "fsinfo block" (file system information block) located at fixed locations

10   on disk. Pointers are used for locating the remaining data. All the data except the

11   fsinfo blocks are collected into files and these files can be written anywhere on the

12   disk.

13

14   •   **Parity checking** – In general, the term "parity checking" refers to an error detection

15   technique that tests the integrity of digital data within a computer system or over a

16   network. Parity bits are checked by comparing them against computed values of

17   parity, which are the XOR of the sets of data bits.

18

19   •   **Parity disk** – In general, the term "parity disk" refers to a separate disk drive that

20   holds parity bits in a disk array, such as four data disks and one parity disk in a

1   volume of a data storage system.

2 • **Parity protected** – In general, the term "parity protected" refers to protection of a

3   collection of data using parity bits. Data is parity protected if it has parity for an

4   entire collection of data. In a preferred embodiment, parity computations can be

5   made across bytes.

6

7 • **Checksum** – In general, the term "checksum" refers to a value used to ensure data is

8   stored or transmitted without error. This value is created by calculating the binary

9   values in a block of data using some algorithm and storing the results with the data or

10   at a separate location. When the data is retrieved from memory, received at the other

11   end of a network, or retrieved from a computer storage system, a new checksum is

12   computed and matched against the existing checksum. A mismatch indicates an error.

13

14   As described herein, the scope and spirit of the invention is not limited to any

15 of the definitions or specific examples shown therein, but is intended to include the most

16 general concepts embodied by these and other terms.

17

18 *System Elements*

19

20   Figure 1 shows a block diagram of a reliable, redundant data storage system

1 including block-appended checksums.

2 A data storage system 100 includes a controller CPU (central processing unit)

3 105, an I/O port 110, a file system 115, a RAID system 125, a disk driver 135, a host/disk

4 adapter 145, a hard disk collection 150, including drive 155, drive 160, drive 165, drive 170

5 and parity drive 175.

6

7 A data storage system 100 is part of a larger computer system. The I/O port

8 110 is connected to the larger computer system in such a way that the controller CPU 105

9 can send data to and from the I/O port 110. The data is written to and read from the hard

10 disk collection 150, including a parity disk 175 in a data storage system 100.

11

12 Unlike other parity systems that may require breaking up the bytes in a block

13 of data or breaking up the block of data itself, each bit in the parity block is computed using

14 the corresponding bits in the data blocks. Thus, if there are four blocks of data, one block

15 would be put on a first drive 155, the second block would be put on drive 160, the third

16 block would be put on drive 165 and the fourth block on drive 170. The parity block is

17 computed using an XOR of the data blocks.

18

19 In a preferred embodiment, the five disk drives 155, 160, 165, 170 and 175 in a

20 RAID system 125 include one or more volumes. A volume is a single file system in a data

1   storage system. Each block has a unique VBN (volume block number) and DBN (disk block

2   number). The VBN specifies the location of a block in a volume. The DBN specifies the

3   location of a block in a disk. Therefore, more than one block can have the same DBN if they

4   are in the same location on different disks. However, only one block can have a given VBN.

5

6   Known data storage systems format hard disks with 512-bytes per sectors.

7   Prior art systems with checksums may include disks formatted with 520-byte sectors

8   comprising 512-bytes of file block data and 8-bytes for a sector checksum. In a preferred

9   embodiment, each disk in a hard disk collection 150 is formatted with 520-bytes per sector.

10   Files are broken into fixed sizes of data known as file blocks. These file blocks are stored in

11   one or more physical sectors of one or more hard disks such as 4,096-bytes that take up 8

12   sectors. With a hard disk formatted to 512-byte sectors, the file block fits into 8 sectors with

13   no extra bytes remaining. With a hard disk formatted for 520-bytes per sector, the 4,096-

14   byte file block fits into the 8 sectors with 64 bytes free for a block-appended checksum. The

15   first 7 sectors contain only file block data while the 8th sector includes the remaining file

16   block data and ends with the 64-byte block-appended checksum. This 520-bytes per sector

17   formatting approach allows the file block and checksum to be written or read in a single

18   operation. The resulting block-appended checksum has an advantage over the prior art

19   sector checksums in a 520-byte formatted hard disk because it can reliably detect sector data

1 corruption such as sector slides, misdirected reads and writes, lost sectors and similar

2 defects.

3 In a preferred embodiment, a series of software and hardware layers is required

4 for reading and writing data between the CPU 105 and the hard disk collection 150. A file

5 system 115 takes a relatively large data file and divides it into a group of file blocks of a

6 given size such as 4,096-bytes. A RAID system stripes these file blocks across a collection

7 of hard disks such as a hard disk collection 150 including four data disks, disk 1 155, disk 2

8 160, disk 3 165 and disk 4 170 plus a parity disk 175 that provides redundancy for the data

9 High performance is accomplished using the RAID system by breaking the group of file

10 blocks into four sub groups and striping these sub groups of blocks in parallel across the data

11 disks. Each file block in a RAID system receives a block-appended checksum of a given

12 size such as 64-bytes. The block-appended checksum is appended to the file block to

13 produce a file block with a block-appended checksum of 4,160-bytes. The block-appended

14 checksum information includes at least: a 4-byte checksum of the data block; a Virtual Block

15 Number (VBN); a Disk Block Number (DBN); and a 4-byte embedded checksum for

16 checking the integrity of the block appended checksum itself. Other embodiments may use

17 other formats of data and algorithms other than Adler's. A sector checksum and a block-

18 appended checksum reliably detect bit flips. However, only a block-appended checksum

19 reliably detects sector data corruption including sector slides, misdirected reads and writes,

20 and lost sectors.

1

2   In a preferred embodiment, the file system 115 allocates a collection of 4,096-

3 byte buffers for each file block when writing a stripe of blocks to the hard disk collection

4 150. Each file block has the same DBN in a stripe provided the hard disk collection 150 is

5 composed of equal sizes of hard disks. Each 4,096-byte file block 120 is written to a 4,096-

6 byte buffer and sent to RAID 125. In RAID 125, each 4,096-byte buffer is appended with

7 64-bytes for a total block size of 4,160 bytes to accommodate the 64-byte block-appended

8 checksum 130. The I/O operations are logged to NVRAM. If the system crashed after this

9 point, the file blocks can be restored upon recovery from the crash by replaying the log of

10 I/O operations from NVRAM. Each 4,096-byte file block plus 64-byte checksum 140 is sent

11 to the disk driver 135. The disk driver 135 creates a scatter/gather list that provides

12 instructions where host/disk adapter 145 should distribute each file block plus 64-byte

13 checksum 140. The collection of buffers and the scatter/gather list are sent to the host/disk

14 adapter 145. The host/disk adapter 145 then writes the stripe of file blocks with the block-

15 appended checksums to the hard disk collection 150 including the four hard disks, disk 1

16 155, disk 2 160, disk 3 165, disk 4 170. The parity data is created from the stripe of file

17 blocks and it is written onto the parity disk 175. A file block with block-appended checksum

18 180 is written to a storage block on disk 1 155 that is composed of 8 sectors of the disk.

19 There is a single operation for writing the file block with appended checksum. The file

20 block data fills all 8 sectors with space remaining in the last part of the last sector to hold the

1   block-appended checksum. When a file is read, each file block and associated block-

2   appended checksum is also done as a single operation. The stored block-appended checksum

3   is compared with a computed block-appended checksum to validate the data. If the stored

4   and computed block-appended checksums are not equivalent, the data has been corrupted

5   and must be rebuilt using the remaining hard disks including the parity disk 175 in the hard

6   disk collection 150.

7

8   *Method of Use*

9

10   Figure 2 shows a flow diagram of a method for writing file blocks with block-

11   appended checksums to a reliable, redundant data storage system.

12

13   A method 200 is performed by the data storage system 100. Although the

14   method 200 is described serially, the steps of the method 200 can be performed by separate

15   elements in conjunction or in parallel, whether asynchronously, in a pipelined manner, or

16   otherwise. There is no particular requirement that the method 200 be performed in the same

17   order in which this description lists the steps, except where so indicated.

18

19   At a flow point 205, the data storage system 100 is ready to perform the

20   method 200 to a file system 115 including writing file blocks and block-appended

1   checksums. In the preferred embodiment the write method 200 requires formatting hard

2   disks to 520 byte sectors.

3

4   At a step 210, the data storage system 100 receives a request from the user to

5   write a file block to the file system 115.

6

7   At a step 215, the data storage system 100 allocates and fills a 4,096-byte

8   buffer with a file block.

9

10   At a step 220, the data storage system 100 sends the filled 4,096-byte buffer to

11   RAID 125.

12

13   At a step 225, the data storage system 100 allocates a 64-byte buffer in RAID

14   125.

15

16   At a flow point 230, the data storage system 100 computes a block-appended

17   checksum for the 4,096-byte file block in the 4,096-byte buffer, fills the 64-byte buffer with

18   the block-appended checksum and appends the 64-byte buffer to the 4,096-byte buffer.

19

20   At step point 235, the data storage system 100 sends the 4,096-byte file block

1 buffer including the 64-byte block-appended checksum buffer to the disk driver 135.

2 At a step point 240, the data storage system 100 creates a scatter/gather list

3 using the disk driver 135 to distribute the 4,096-byte file block with appended checksum to a

4 group of sectors making up a storage block on one or more of the disks in the hard disk

5 collection 150.

6

7 At a step 245, the data storage system 100 sends the 4,096-byte buffer,

8 including the appended 64-byte buffer and the scatter/gather list to the host/disk adapter 145.

9

10 At a step 250, the data storage system 100 writes the file block with the block-

11 appended checksum to a storage block in a single operation.

12

13 At a step 255, the data storage system 100 completes writing to one or more of

14 the hard disks in the hard disk collection 150.

15 At a step 260, the data storage system 100 frees up the 64-byte buffer in RAID

16 125.

17

18 At a flow point 265, the data storage system 100 has succeeded or failed at

19 writing a file to the file system.

20

1   Figure 3 shows a flow diagram of a method for reading data from data storage

2   blocks including file blocks with block-appended checksums in a reliable, redundant data

3   storage system.

4

5   A read method 300 is performed by the data storage system 100. Although the

6   read method 300 is described serially, the steps of the read method 300 can be performed by

7   separate elements in conjunction or in parallel, whether asynchronously, in a pipelined

8   manner, or otherwise. There is no particular requirement that the read method 300 be

9   performed in the same order, in which this description lists the steps, except where so

10   indicated.

11

12   At a flow point 305, the data storage system 100 is ready for requests to read

13   file blocks from a file system 115, including reading file blocks and block-appended

14   checksums.

15   At a step 310, the data storage system 100 receives a request from the user to

16   read a file block to the file system. 115.

17

18   At a step 315, the data storage system 100 allocates a 4,096-byte buffer for a

19   file block.

20

1   At a step 320, the data storage system 100 sends the empty 4,096-byte buffer to

2 RAID 125.

3

4   At a step 325, the data storage system 100 allocates a 64-byte buffer in RAID

5 125 and appends it to the 4,096-byte buffer.

6

7   At a flow point 330, the data storage system 100 sends the 4,096-byte file

8 block buffer with the 64-byte block-appended checksum buffer to the disk driver 135.

9

10   At a step point 335, the data storage system 100 creates a scatter/gather list in

11 the disk driver 135 to collect the 4,096-byte file block from a group of sectors making up a

12 storage block on one or more of the disks in the hard disk collection 150.

13

14   At a step 340, the data storage system 100 sends the 4,096-byte buffer with the

15 appended 64-byte buffer along with the scatter/gather list to the host/disk adapter 145.

16

17   At a step 345, the data storage system 100 reads the file block with the block-

18 appended checksum from a storage block in a single operation.

19

20   At a step 350, the data storage system 100 completes reading the file block

1   with the block-appended checksum from one or more of the hard disks collection 150.

2   At a step 355, the data storage system 100 computes the block-appended

3   checksum for the file block and compares it to the appended block-appended checksum to

4   verify the file block.

5

6   At a step 360, the data storage system 100 frees up the 64-byte buffer in RAID

7   125.

8

9   At a flow point 365, the data storage system 100 has succeeded or failed at

10   reading a file block from the file system 115.

11

12   *Alternative Embodiments*

13

14   Although preferred embodiments are disclosed herein, many variations

15   are possible which remain within the concept, scope, and spirit of the invention, and these

16   variations would become clear to those skilled in the art after perusal of this application.